SC-703 Desarrollo de Aplicaciones Móviles

I CO 2022
SEMANA 2-3
Room/ SQLite / Mejorando Diseños

• Agenda

- Configuración del proyecto
- ROOM / SQLite
- Mejorando las vistas

fidélitas
INGENIERÍA EN SISTEMAS DE COMPUTACIÓN

• AJUSTAR EL PROYECTO DE LA SEMANA PASADA (SUBIDO POR EL PROFESORI

Si desea tomar el proyecto desarrollado por el profesor y continuarlo esta semana

En semana 02 está el proyecto subido. Se debe descargar y utilizar el “Google-services.json” de su cuenta de Firebase

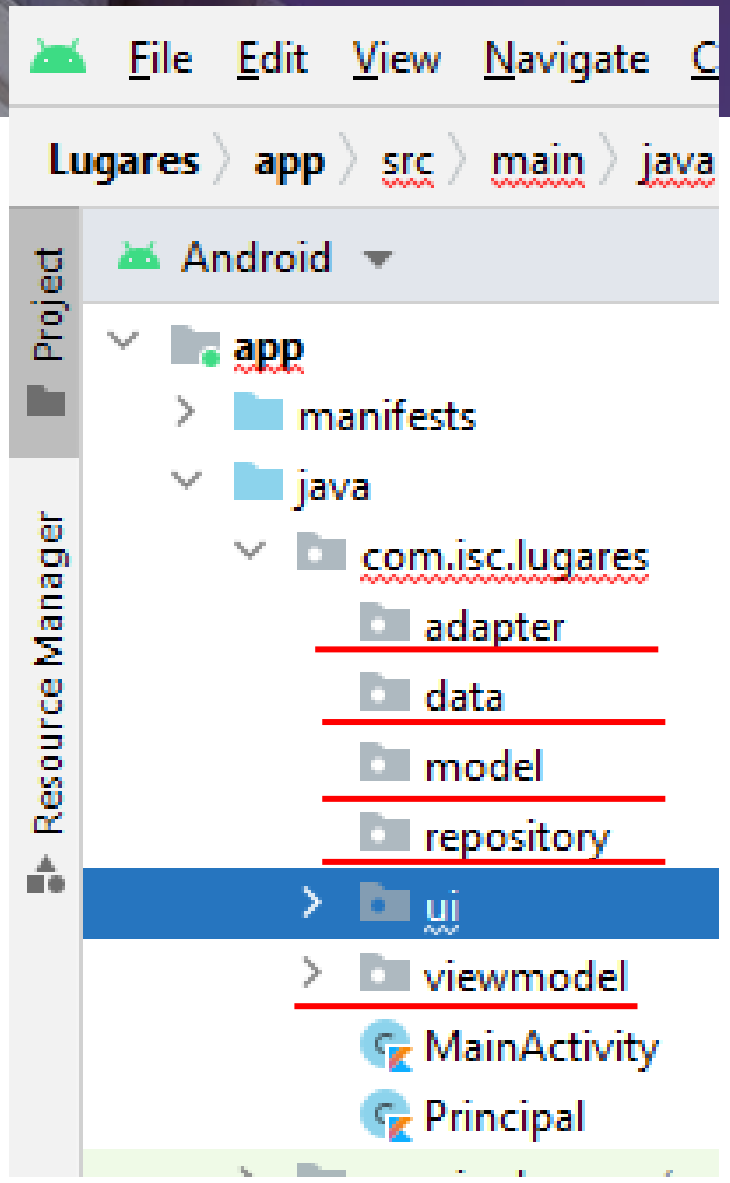
Luego de esto ya está listo para ver la clase de hoy

- **MEJORANDO LA ARQUITECTURA DEL PROYECTO**

fidélitas Universidad

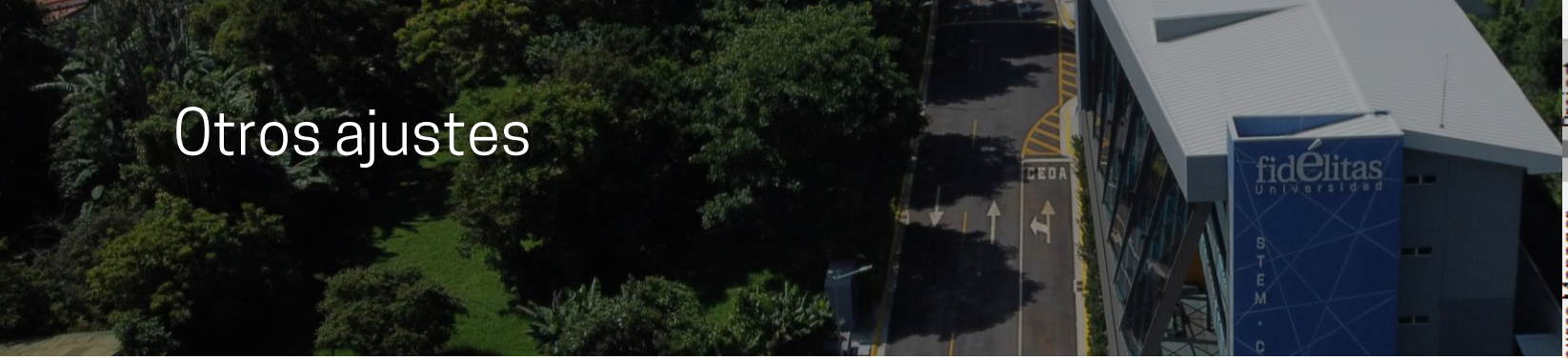
- **SE CREAN LOS SIGUIENTES PAQUETES EN EL PROYECTO LUGARES**

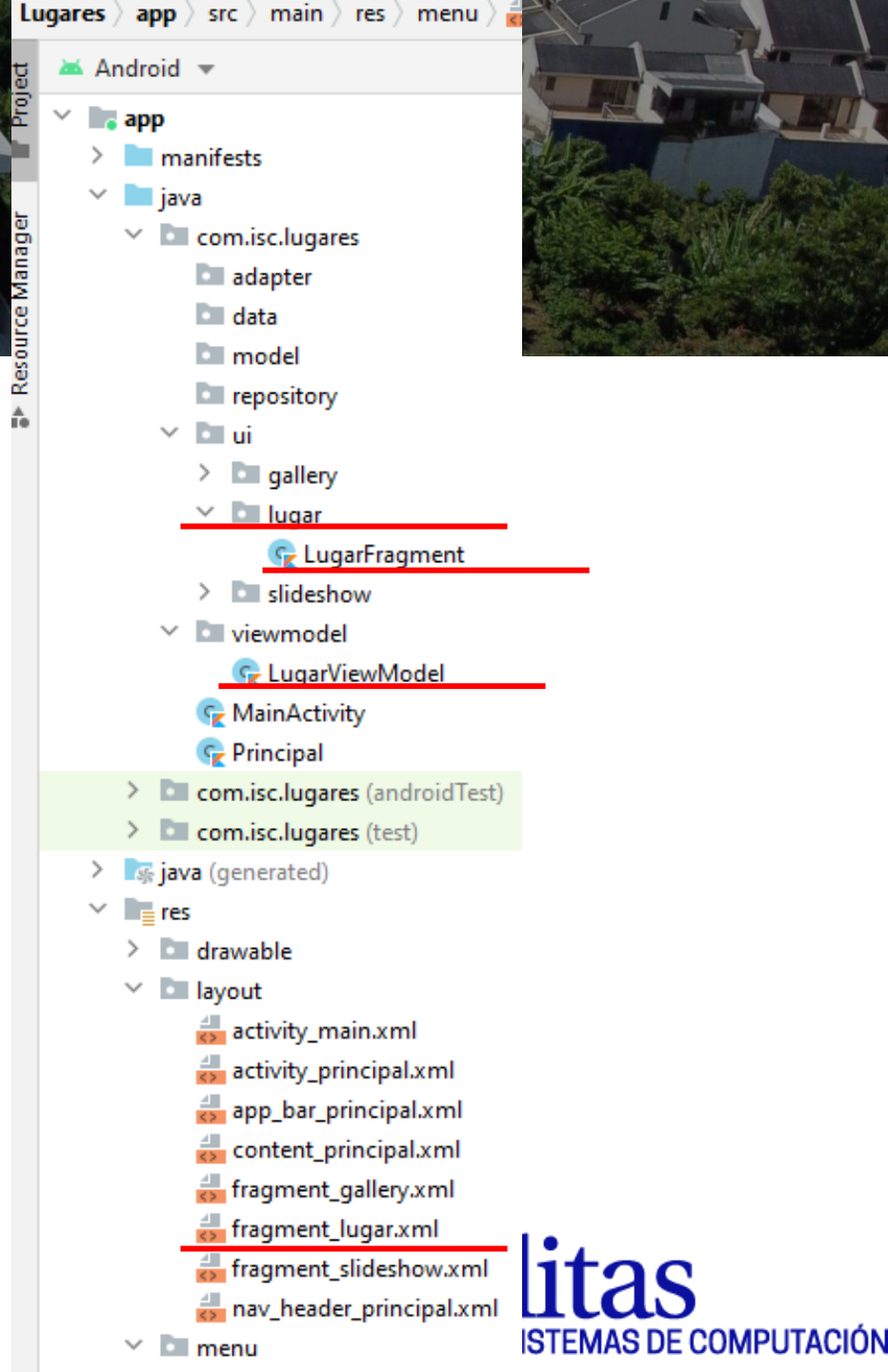
adapter
data
model
repository
viewmodel

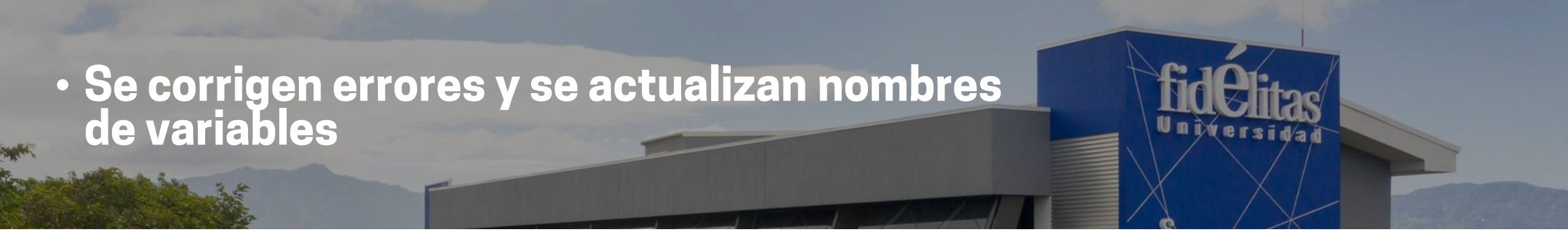


Otros ajustes

1. Renombramos el paquete:
   - home → lugar
2. Renombramos las clases:
   - HomeFragment → LugarFragment
   - HomeViewModel → LugarViewModel
3. Movemos:
   - LugarViewModel al paquete viewmodel
4. Renombramos el archivo:
   - fragment_home.xml → fragment_lugar.xml







- **Se corrigen errores y se actualizan nombres de variables**

LugarFragment.kt

```
import androidx.lifecycle.ViewModelProvider
import com.isc.lugares.databinding.FragmentHomeBinding
import com.isc.lugares.viewmodel.LugarViewModel

class LugarFragment : Fragment() {

    private lateinit var homeViewModel: LugarViewModel
    private var _binding: FragmentHomeBinding? = null

    // This property is only valid between onCreateView and
    // onDestroyView.
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        homeViewModel =
            ViewModelProvider( owner: this).get(LugarViewModel::class.java)

        _binding = FragmentHomeBinding.inflate(inflater, container, false)
        val root: View = binding.root
```

LugarFragment.kt

```
import com.isc.lugares.databinding.FragmentLugarBinding
import com.isc.lugares.viewmodel.LugarViewModel

class LugarFragment : Fragment() {

    private lateinit var lugarViewModel: LugarViewModel
    private var _binding: FragmentLugarBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        lugarViewModel =
            ViewModelProvider( owner: this).get(LugarViewModel::class.java)

        _binding = FragmentLugarBinding.inflate(inflater, container, attach
        val root: View = binding.root

        val textView: TextView = binding.textHome
        lugarViewModel.text.observe(viewLifecycleOwner, Observer { it: String!
            textView.text = it
        })
```

- **Actualizamos archivo string.xml**

strings.xml

Edit translations for all locales in the translations editor.

```
1  <resources>
2      <string name="app_name">Lugares</string>
3      <string name="msg_datos">Requiere texto en los ca
4      <string name="msg_fallo">Falló la autenticación..
5      <string name="title_activity_principal">Principal
6      <string name="navigation_drawer_open">Open naviga
7      <string name="navigation_drawer_close">Close navi
8      <string name="nav_header_title">Android Studio</s
9      <string name="nav_header_subtitle">android.studio(
10     <string name="nav_header_desc">Navigation header<
11     <string name="action_settings">Settings</string>
12     <string name="menu_lugar">Lugares</string>
13     <string name="menu_gallery">Gallery</string>
14     <string name="menu_slideshow">Slideshow</string>
15 </resources>
```

- **Actualizamos el archivo mobile_navigation.xmls**

mobile_navigation.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <navigation xmlns:android="http://schemas.android.com/apk/res/a
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/mobile_navigation"
6      app:startDestination="@+id/nav_lugar">
7
8      <fragment
9          android:id="@+id/nav_lugar"
10         android:name="com.isc.lugares.ui.lugar.LugarFragment"
11         android:label="@string/menu_lugar"
12         tools:layout="@layout/fragment_lugar" />
13
```

- Se modifica el menú activity_main_Drawer.xml

activity_main_drawer.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/
3      xmlns:tools="http://schemas.android.com/tools"
4      tools:showIn="navigation_view">
5
6      <group android:checkableBehavior="single">
7          <item
8              android:id="@+id/nav_lugar"
9              android:icon="@drawable/ic_menu_camera"
10             android:title="@string/menu_lugar" />
```

- Se modifica la clase Principal.kt

Principal.kt

```
32          val navController = findNavController(R.id.nav_
33          // Passing each menu ID as a set of Ids because
34          // menu should be considered as top level desti
35          appBarConfiguration = AppBarConfiguration(
36              setOf(
37                  R.id.nav_lugar, R.id.nav_gallery, R.id.
38              ), drawerLayout
39          )
40          setupActionBarWithNavController(navController, 
41          navView.setupWithNavController(navController)
42      }
```

- Ya se tiene el proyecto listo en arquitectura, ahora se prepara para nuevo material

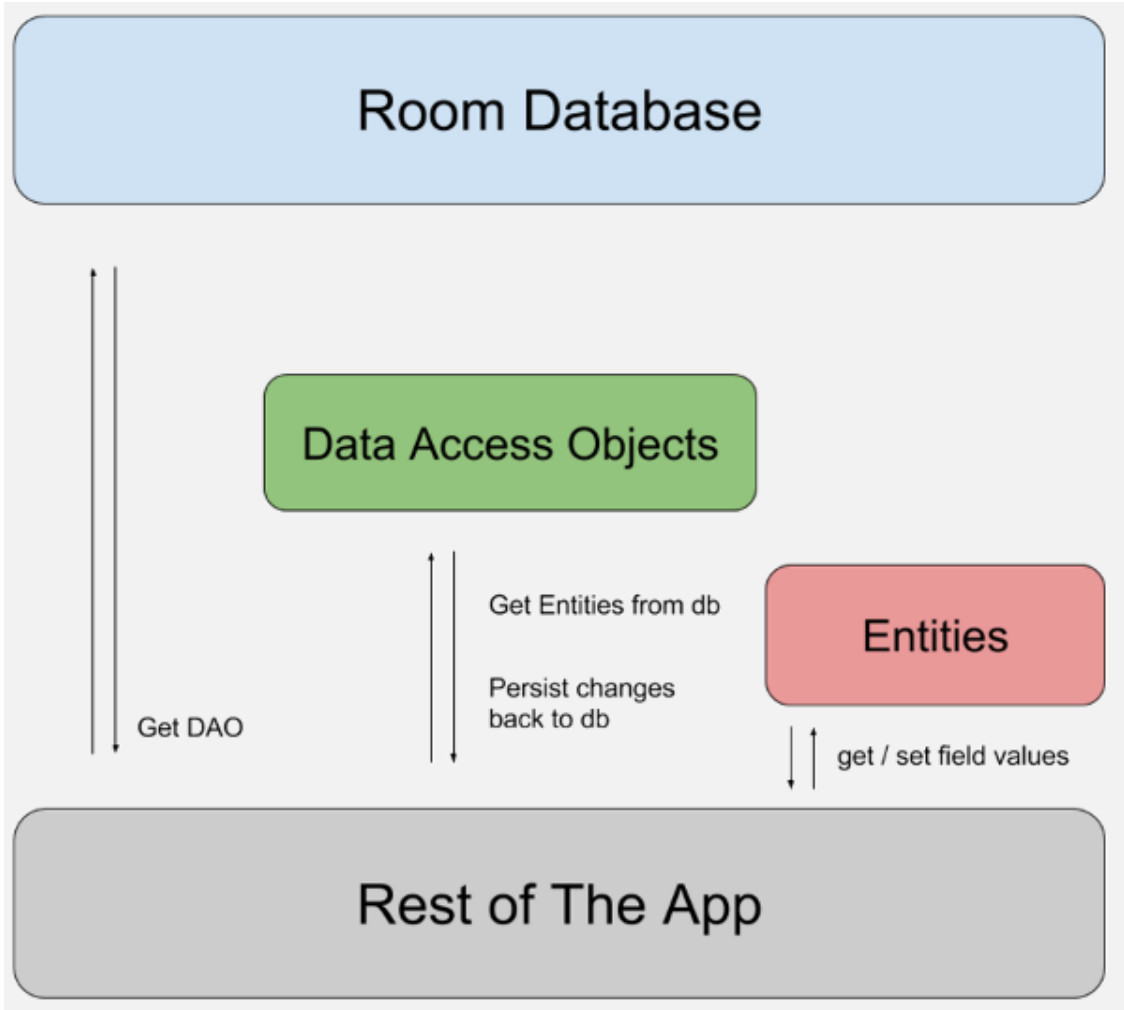
fidélitas Universidad

Uso de Room sobre SQLite
Base de datos local en el celular

- Verificación de las sentencias SQL en tiempo de compilación
- Menor repetición de código (boilerplate code)
- Fácilmente integrable con otras componentes de la arquietectura

fidélitas
INGENIERÍA EN SISTEMAS DE COMPUTACIÓN

- **Room**



- **Anotaciones**

@Entity: representa una tabla dentro de la base de datos, es definida dentro de una clase, no tienen lógica, sólo atributos

@Dao: Es la parte responsable de definir las funciones para accesar la base de datos, donde se colocan las sentencias.

@Database: contiene la definición general de la base de datos, lo de más bajo nivel.

- **Actualización de archivo build.gradle**

fidélitas
Universidad

- **Actualizamos el build.gradle (project)**

build.gradle (Lugares)

You can use the Project Structure dialog to view and edit your project configuration

```
4        google()
5        mavenCentral()
6    }
7    dependencies {
8        classpath "com.android.tools.build:gradle:7.0.2"
9        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31"
10       classpath 'com.google.gms:google-services:4.3.10'
11
12       //Para poder pasar argumentos en la navegación entre fragmentos
13       classpath "androidx.navigation:navigation-safe-args-gradle-plugin:2.3.5"
14
```

• build.gradle (module)

build.gradle (:app)

You can use the Project Structure dialog to view and edit your project configuration

```
1   plugins {
2       id 'com.android.application'
3       id 'kotlin-android'
4       id 'com.google.gms.google-services'
5       id 'kotlin-parcelize'
6   }
7   apply plugin: 'kotlin-kapt'
8   apply plugin: "androidx.navigation.safeargs"  //Para pasar los parámetros...
9
10  //Para generar un enlace directo entre la vista (xml) y el código...
11  android.buildFeatures.viewBinding true
```

- Contin
archiv

build.gradle (:app)

You can use the Project Structure dialog to view and edit your project configuration

```
58      androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
59
60      //Se agregan por @Entity de Lugar...
61      annotationProcessor 'androidx.room:room-compiler:2.3.0'
62      kapt "androidx.room:room-compiler:2.3.0"
63      implementation "androidx.room:room-ktx:2.3.0"
64
65      //Navegacion entre fragmentos
66      implementation "androidx.navigation:navigation-fragment-ktx:2.3.5"
67      implementation "androidx.navigation:navigation-ui-ktx:2.3.5"
68      implementation "androidx.navigation:navigation-dynamic-features-fragment:2.3.5"
69
70      //Para clases más adelante...
71      // Esto es para la parte de gps y camara
72      implementation 'com.google.android.gms:play-services-location:18.0.0'
73      implementation 'androidx.camera:camera-camera2:1.1.0-alpha09'
74
75      // para usar las imagenes en el recyclerview desde firestore
76      implementation 'com.github.bumptech.glide:glide:4.12.0'
77      //Para usar redondeo de imagenes y demás
78      implementation 'jp.wasabeef:glide-transformations:4.1.0'
79  }
```


PUTACIÓN

- Se inicia con el proceso para almacenar la información de los lugares

fidélitas
Universidad

- ¿Qué debemos almacenar del lugar?
  ¿Cómo será la clase Lugar.kt

fidélitas
Universidad

- Se crea la clase Lugar.kt dentro del paque model

Lugar.kt

```kotlin
package com.isc.lugares.model

import android.os.Parcelable
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
import kotlinx.parcelize.Parcelize

@Parcelize
@Entity(tableName = "lugar")
data class Lugar(
    @PrimaryKey(autoGenerate = true)
    val id: Int,
    @ColumnInfo(name = "nombre")
    val nombre: String?,
    @ColumnInfo(name = "correo")
    val correo: String?,
    @ColumnInfo(name = "telefono")
    val telefono: String?,
    @ColumnInfo(name = "latitud")
    val latitud: Double?,
    @ColumnInfo(name = "longitud")
    val longitud: Double?,
    @ColumnInfo(name = "altura")
    val altura: Int?,
    @ColumnInfo(name = "rutaAudio")
    val rutaAudio: String?,
    @ColumnInfo(name = "rutaImagen")
    val rutaImagen: String?
) : Parcelable
```

- Se crea la interface LugarDao en el paquete data

LugarDao.kt

```
1   package com.isc.lugares.data
2
3   import androidx.lifecycle.LiveData
4   import androidx.room.*
5   import com.isc.lugares.model.Lugar
6
7   @Dao
8   interface LugarDao {
9       @Query( value: "SELECT * FROM Lugar")
10      fun getAllData() : LiveData<List<Lugar>>
11
12      @Insert(onConflict = OnConflictStrategy.IGNORE)
13      suspend fun addLugar(lugar: Lugar)
14
15      @Update
16      suspend fun updateLugar(lugar: Lugar)
17
18      @Delete
19      suspend fun deleteLugar(lugar: Lugar)
20  }
```

- Se crea la clase Luga
en el paquete data

LugarDatabase.kt

```kotlin
package com.isc.lugares.data

import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import com.isc.lugares.model.Lugar

@Database(entities=[Lugar::class], version = 1,exportSchema = false)
abstract class LugarDatabase: RoomDatabase() {
    abstract fun lugarDao(): LugarDao

    companion object {
        @Volatile
        private var INSTANCE: LugarDatabase? = null

        fun getDatabase(context: android.content.Context): LugarDatabase {
            val tempInstance = INSTANCE
            if (tempInstance != null) {
                return tempInstance
            }
            synchronized( lock: this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    LugarDatabase::class.java,
                     name: "lugar_database"
                ).build()
                INSTANCE = instance
                return instance
            }
        }
    }
```

- **Se crea la clase LugarRespository en el paquete repository**

LugarRepository.kt

```kotlin
package com.isc.lugares.repository

import androidx.lifecycle.LiveData
import com.isc.lugares.data.LugarDao
import com.isc.lugares.model.Lugar

class LugarRepository(private val lugarDao: LugarDao) {
    val getAllData: LiveData<List<Lugar>> = lugarDao.getAllData()

    suspend fun addLugar(lugar: Lugar) {
        lugarDao.addLugar(lugar)
    }

    suspend fun updateLugar(lugar: Lugar) {
        lugarDao.updateLugar(lugar)
    }

    suspend fun deleteLugar(lugar: Lugar) {
        lugarDao.deleteLugar(lugar)
    }
}
```

Un repositorio es una clases abstracta para acceder a múltiples fuentes de datos. No es parte de la Librería de Arquitectura de Componentes, se sugiere como una buena práctica para separan la codificación y la arquitectura.

- **Se crea la clase LugarViewModel en el paquete viewmodel**

LugarViewModel.kt

```
11  class LugarViewModel(application: Application): AndroidViewModel(application) {
12      val getAllData: LiveData<List<Lugar>>
13      private val repository: LugarRepository
14      init {
15          val lugarDao = LugarDatabase.getDatabase(application).lugarDao()
16          repository = LugarRepository(lugarDao)
17          getAllData = repository.getAllData
18      }
19      fun addLugar(lugar: Lugar) {
20          viewModelScope.launch(Dispatchers.IO) { repository.addLugar(lugar) }
21      }
22      fun deleteLugar(lugar: Lugar) {
23          viewModelScope.launch(Dispatchers.IO) { repository.deleteLugar(lugar) }
24      }
25      fun updateLugar(lugar: Lugar) {
26          viewModelScope.launch(Dispatchers.IO) { repository.updateLugar(lugar) }
27      }
28  }
```

La función de ViewModel es proveer la información hacia al Interfaz del usuario (UI) y mantenerse a pesar de los cambios de la configuración. El modelo actúa como un centro de comunicación entre el repositorio y la UI

• Los imports de LugarViewModel

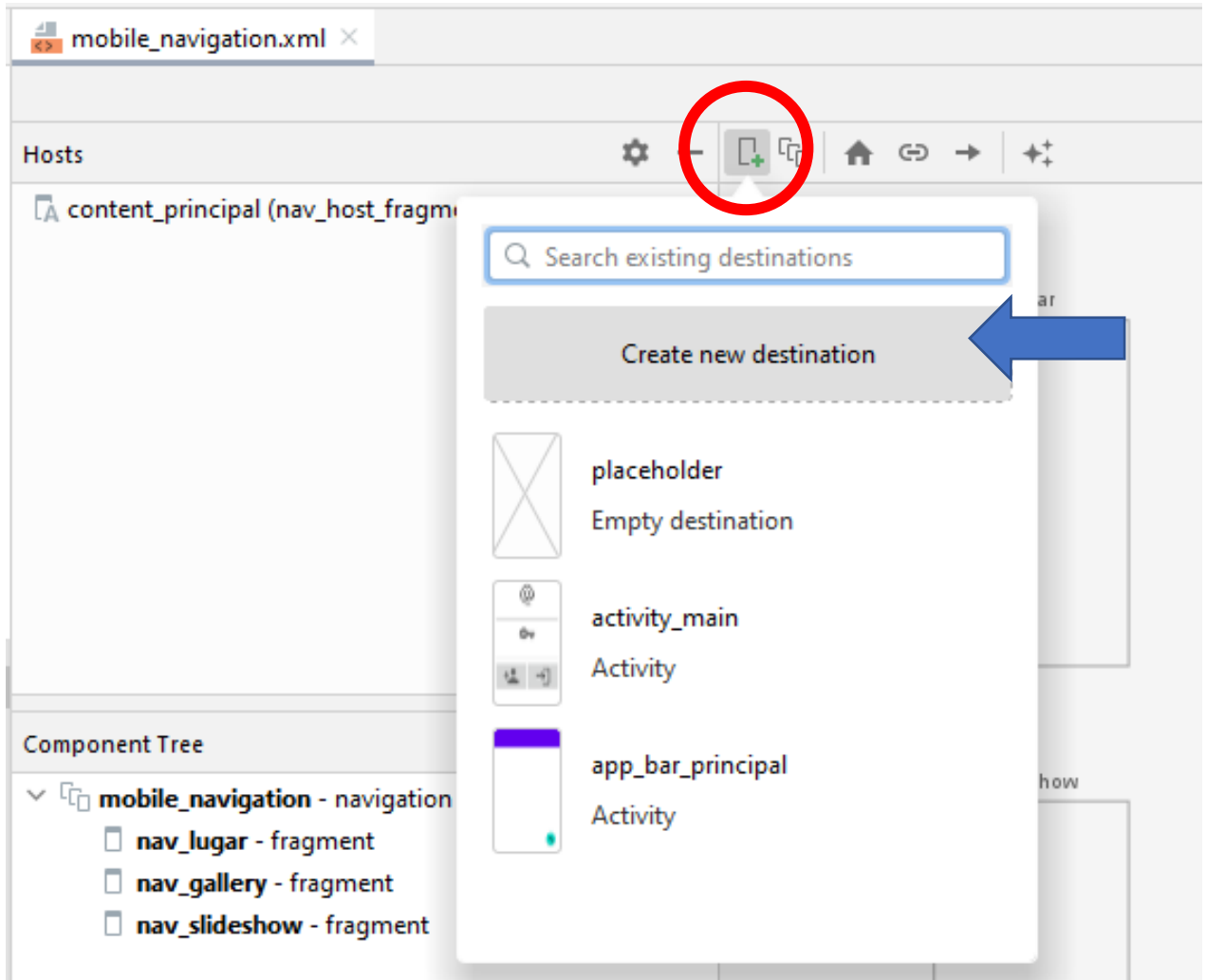
LugarViewModel.kt

```kotlin
package com.isc.lugares.viewmodel
import android.app.Application
import androidx.lifecycle.AndroidViewModel
import androidx.lifecycle.LiveData
import androidx.lifecycle.viewModelScope
import com.isc.lugares.data.LugarDatabase
import com.isc.lugares.model.Lugar
import com.isc.lugares.repository.LugarRepository
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
class LugarViewModel(application: Application): AndroidViewModel(application) {
    val getAllData: LiveData<List<Lugar>>
    private val repository: LugarRepository
    init {
        val lugarDao = LugarDatabase.getDatabase(application).lugarDao()
        repository = LugarRepository(lugarDao)
        getAllData = repository.getAllData
    }
```

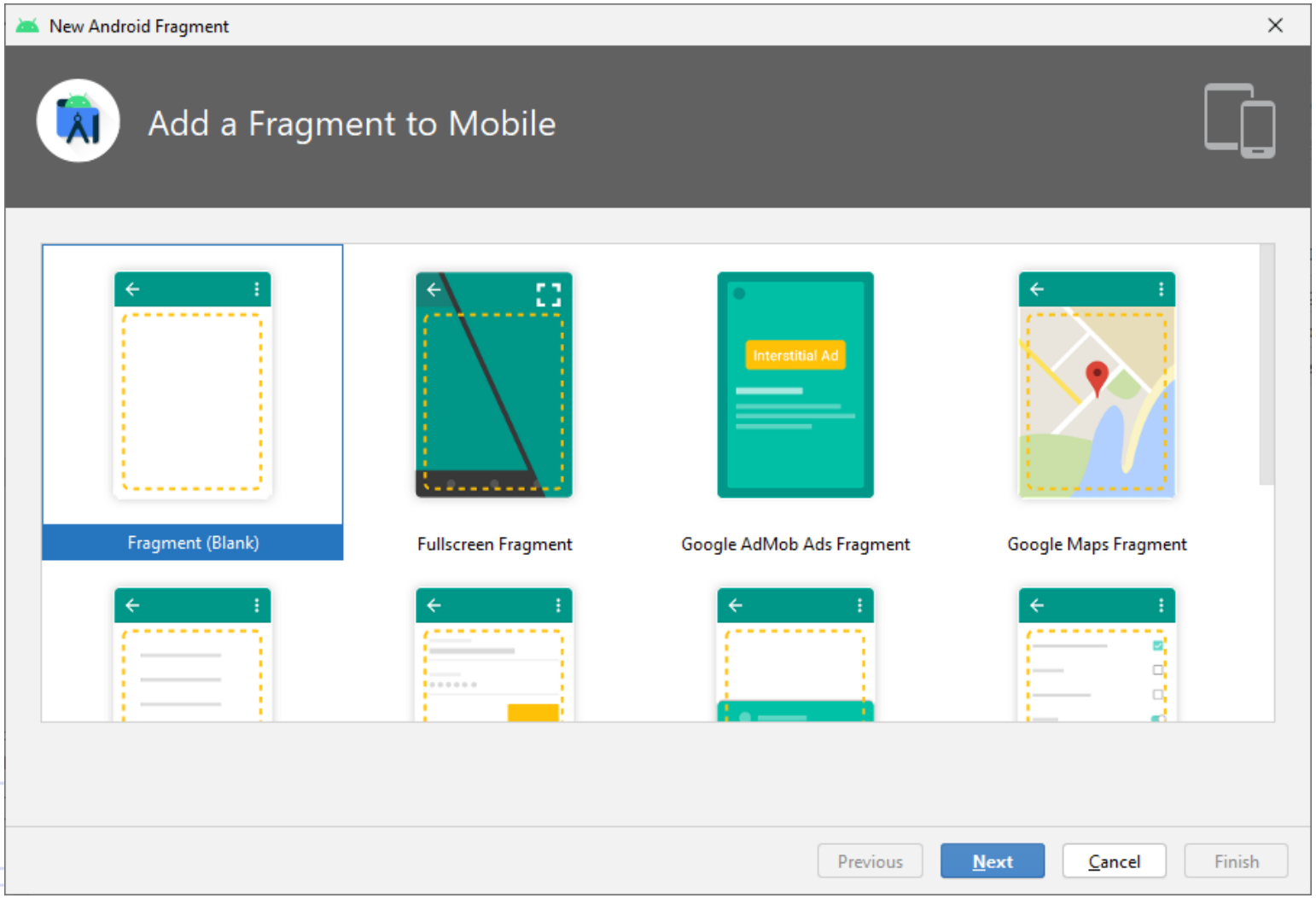
- Se crean los fragmentos y la navegación hacia estos

fidélitas Universidad

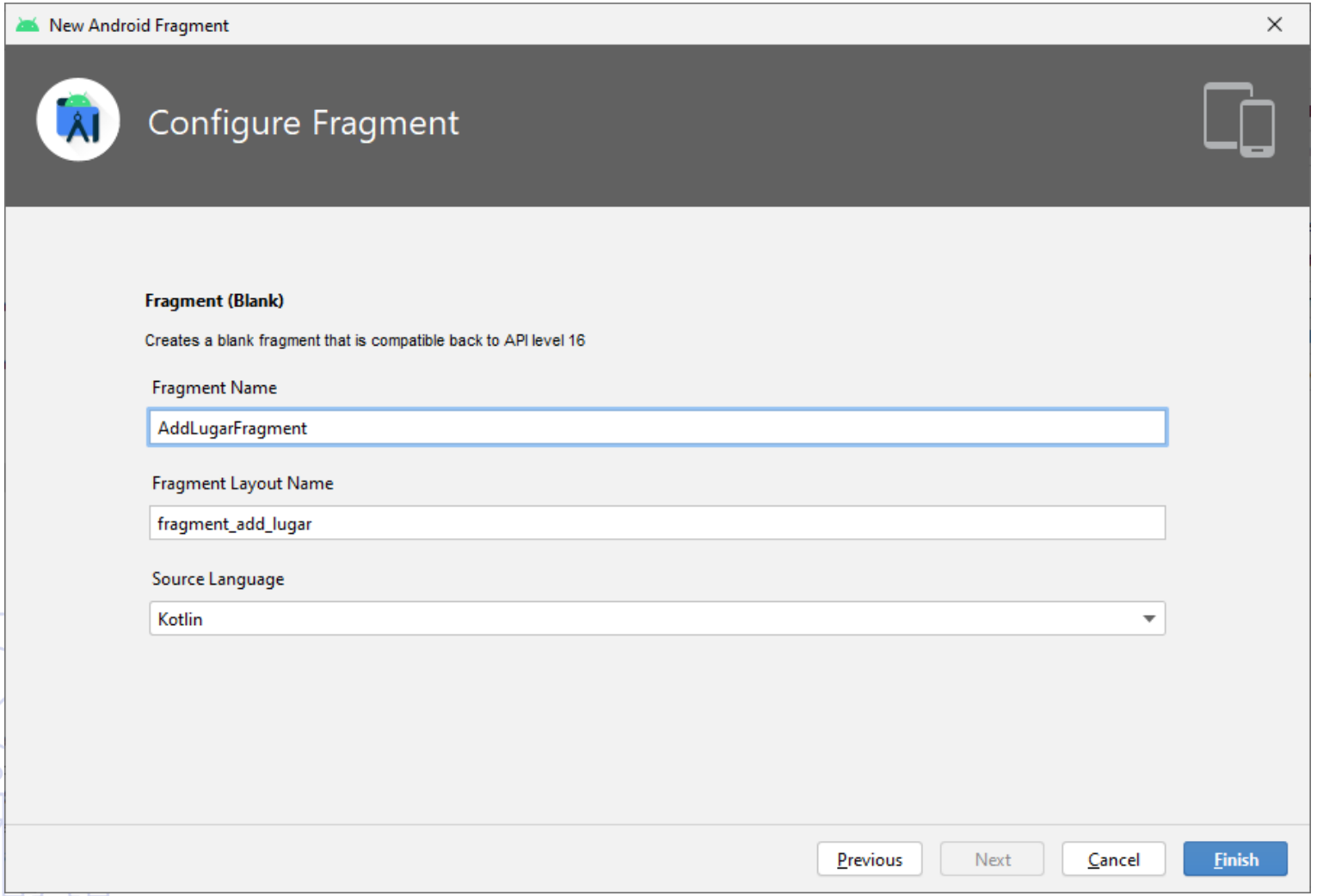
- **Se da clic en un nuevo destino**



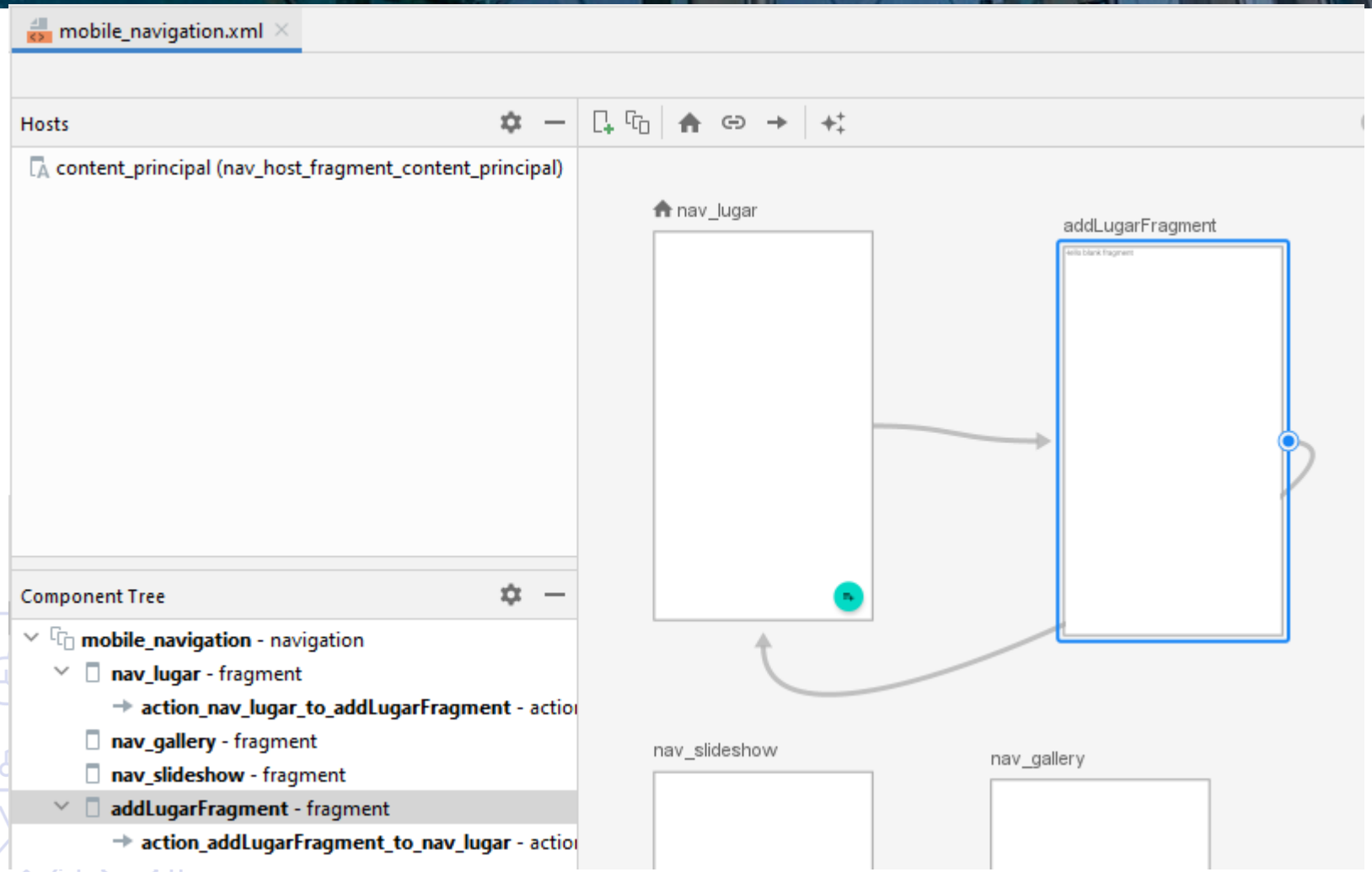
- Se crea un fragmento en blanco



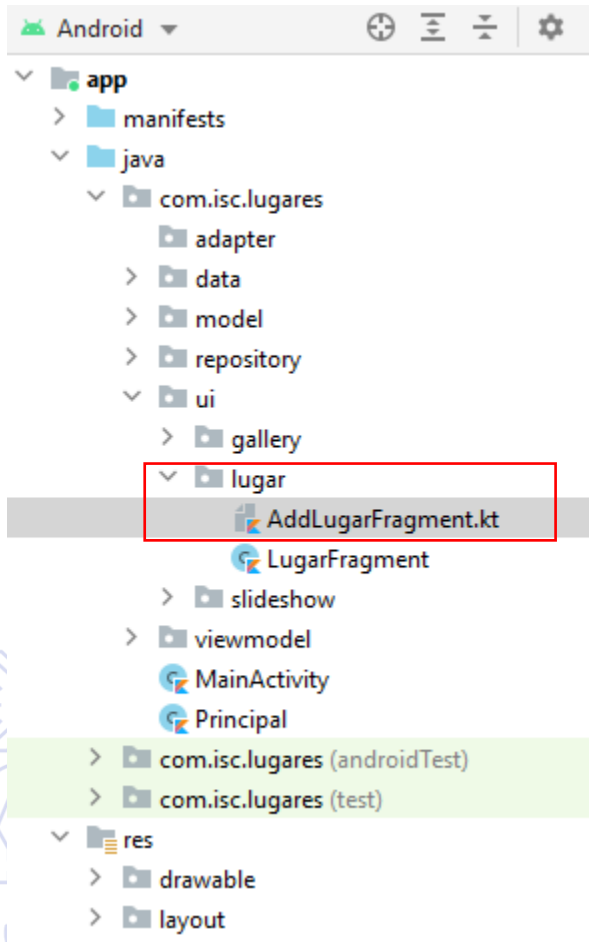
- Se define como AddLugarFragment



- Se definen las navegaciones por medio de los bordes



- Se mueve AddLugarFragment.kt y se modifica mobile_navigation.xml



mobile_navigation.xml | AddLugarFragment.kt

Code

```
21      android:label="Gallery"
22      tools:layout="@layout/fragment_gallery" />
23
24  <fragment
25      android:id="@+id/nav_slideshow"
26      android:name="com.isc.lugares.ui.slideshow.SlideshowFragment"
27      android:label="Slideshow"
28      tools:layout="@layout/fragment_slideshow" />
29  <fragment
30      android:id="@+id/addLugarFragment"
31      android:name="com.isc.lugares.ui.lugar.AddLugarFragment"
32      android:label="fragment_add_lugar"
33      tools:layout="@layout/fragment_add_lugar" >
```

- **LugarFragment.kt se ajusta así**

LugarFragment.kt

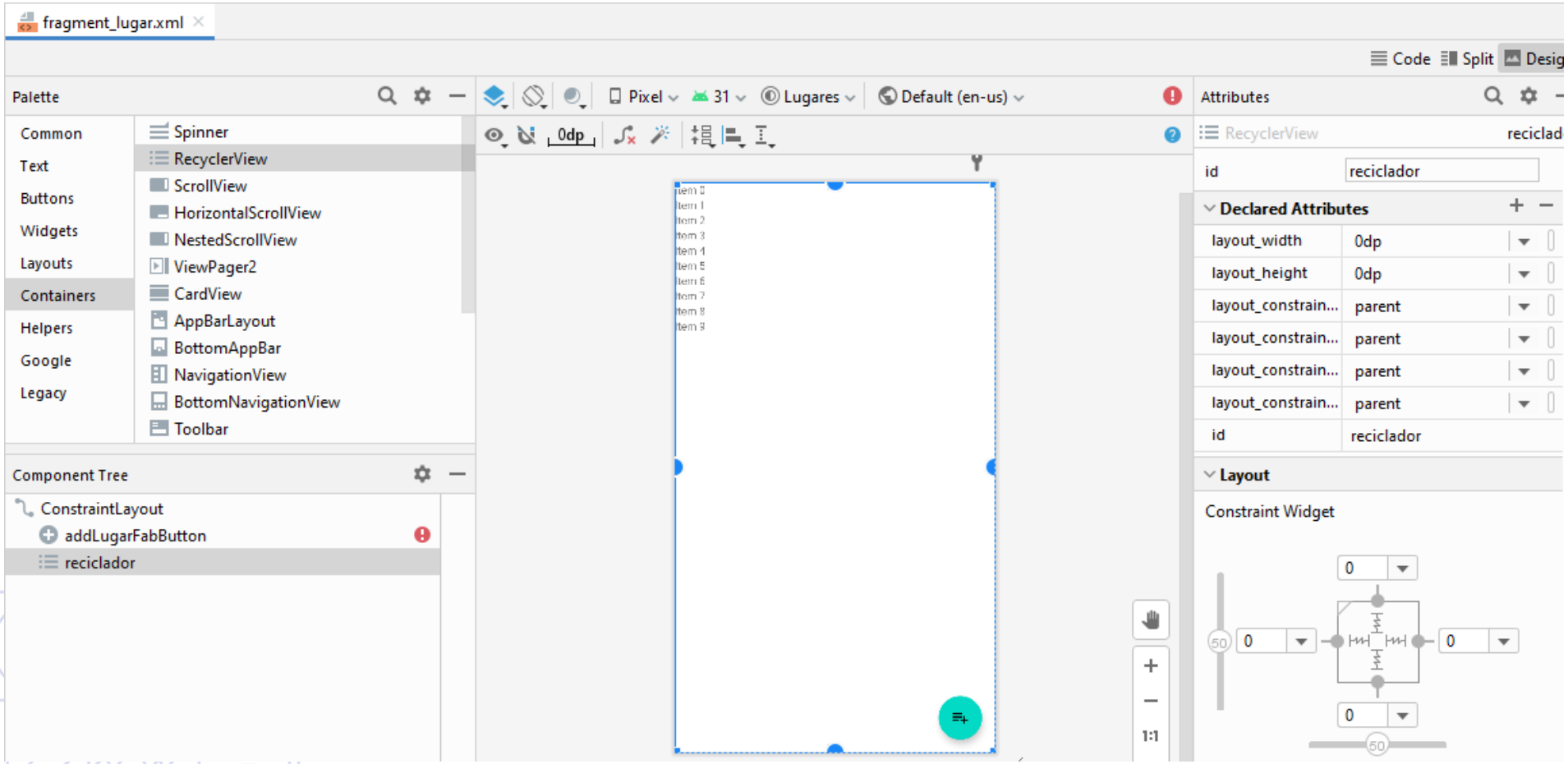
```
import com.isc.lugares.databinding.FragmentLugarBinding
import com.isc.lugares.viewmodel.LugarViewModel
class LugarFragment : Fragment() {
    private lateinit var lugarViewModel: LugarViewModel
    private var _binding: FragmentLugarBinding? = null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        lugarViewModel =
            ViewModelProvider( owner: this).get(LugarViewModel::class.java)

        _binding = FragmentLugarBinding.inflate(inflater, container,  attachToParent: false)

        return binding.root
    }
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

• Así queda la clase AddLugarFragment

AddLugarFragment.kt

```
3   import android.os.Bundle
4   import androidx.fragment.app.Fragment
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import com.isc.lugares.databinding.FragmentAddLugarBinding
9
10  class AddLugarFragment : Fragment() {
11      private var _binding: FragmentAddLugarBinding?=null
12      private val binding get() = _binding!!
13
14      override fun onCreateView(
15          inflater: LayoutInflater, container: ViewGroup?,
16          savedInstanceState: Bundle?
17      ): View {
18          _binding = FragmentAddLugarBinding.inflate(layoutInflater,container, attachToParent: false)
19
20          return binding.root
21      }
22  }
```

• Así debe quedar fragment_lugar.xml



fidélitas
INGENIERÍA EN SISTEMAS DE COMPUTACIÓN

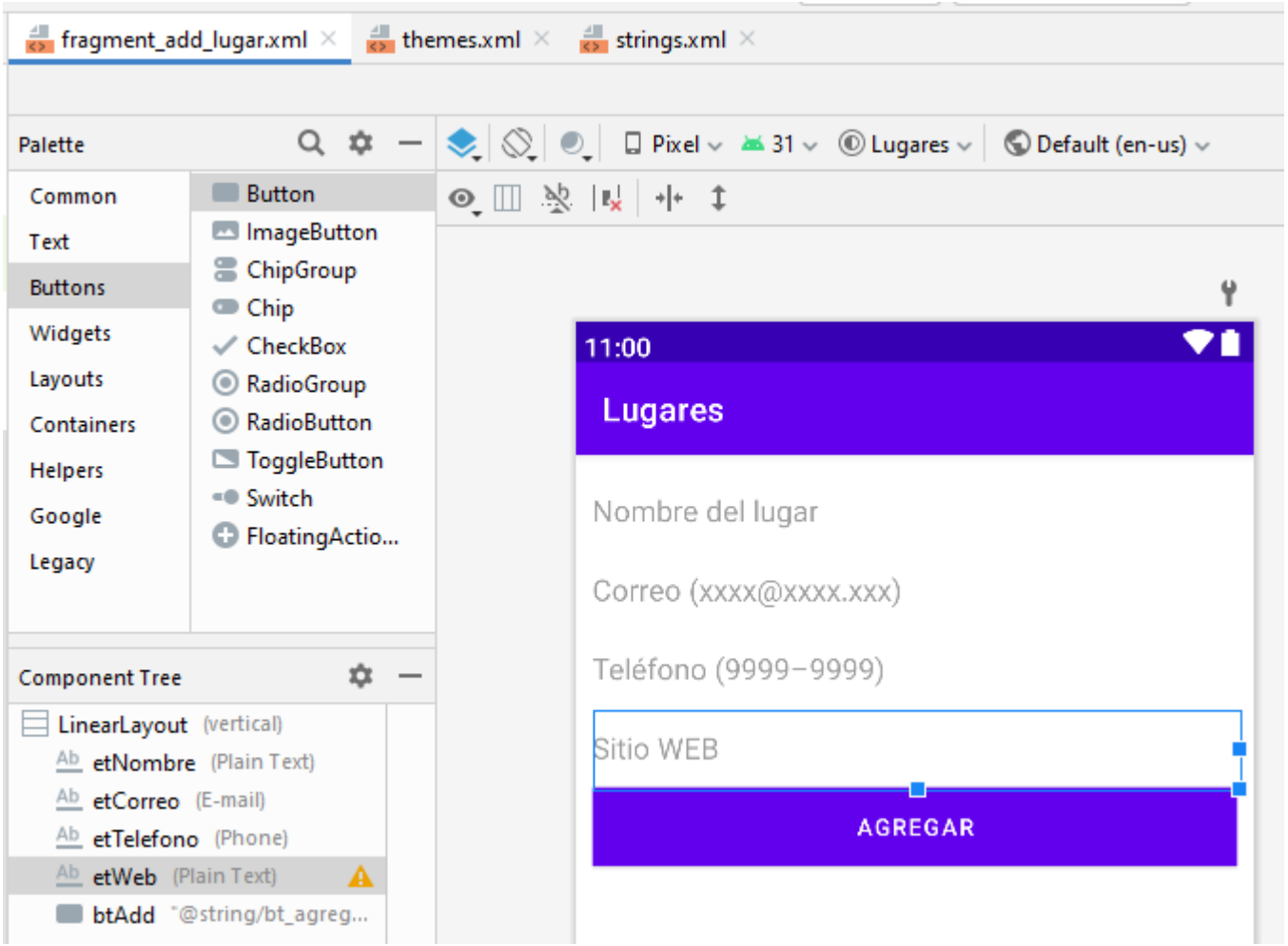
- Se agregan 5 string en el archivo strings.xml

strings.xml

Edit translations for all locales in the translations editor.

```
11      <string name="action_settings">Settings</string>
12      <string name="menu_lugar">Lugares</string>
13      <string name="menu_gallery">Gallery</string>
14      <string name="menu_slideshow">Slideshow</string>
15
16      <string name="msg_nombre">Nombre del lugar</string>
17      <string name="msg_Correo">Correo (xxxx@xxxx.xxx)</string>
18      <string name="msg_Telefono">Teléfono (9999-9999)</string>
19      <string name="msg_web">Sitio WEB</string>
20      <string name="bt_Agregar">Agregar</string>
```

• Así debe quedar fragment_add_lugar.xml



fidélitas
INGENIERÍA EN SISTEMAS DE COMPUTACIÓN

- **Actualizamos Lugar_Fragment.kt**

LugarFragment.kt

```
19          savedInstanceState: Bundle?
20      ): View {
21          lugarViewModel =
22              ViewModelProvider( owner: this).get(LugarViewModel::class.java)
23          _binding = FragmentLugarBinding.inflate(inflater, container,  attachToParent: false)
24
25          binding.addLugarFabButton.setOnClickListener {  it: View!
26              findNavController().navigate(R.id.action_nav_lugar_to_addLugarFragment)
27          }
28
29          return binding.root
30      }
```

• Modificamos AddLugarFragment.kt

AddLugarFragment.kt

```
import androidx.lifecycle.ViewModelProvider
import com.isc.lugares.databinding.FragmentAddLugarBinding
import com.isc.lugares.viewmodel.LugarViewModel

class AddLugarFragment : Fragment() {
    private var _binding: FragmentAddLugarBinding?=null
    private val binding get() = _binding!!

    private lateinit var lugarViewModel: LugarViewModel

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentAddLugarBinding.inflate(layoutInflater,container, attachToParent: false)

        lugarViewModel = ViewModelProvider( owner: this).get(LugarViewModel::class.java)

        binding.btAddLugar.setOnClickListener { insertaLugar() }

        return binding.root
    }
```

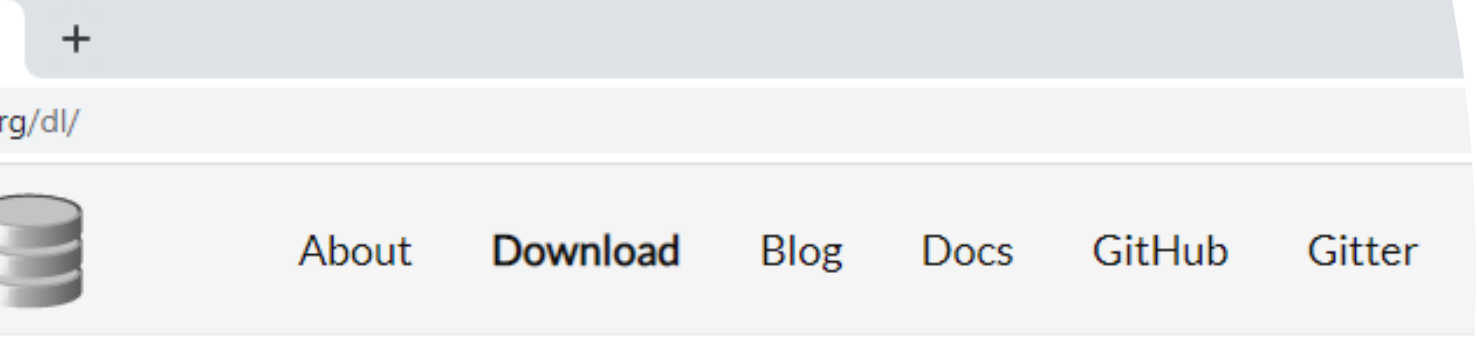
- **Se actualiza AddLugar.Fragment.kt**

AddLugarFragment.kt

```
35      private fun insertaLugar() {
36          val nombre= binding.etNombre.text.toString()
37          val correo= binding.etCorreo.text.toString()
38          val telefono = binding.etTelefono.text.toString()
39          val web=binding.etWeb.text.toString()
40          if (validos(nombre,correo,telefono,web)) {
41              val lugar= Lugar( id: 0,nombre,correo,telefono,web, longitud: 0.0, latitud: 0.0, altura: 0, rutaAudio: "", rutaImagen: "")
42              lugarViewModel.addLugar(lugar)
43              Toast.makeText(requireContext(),getString(R.string.msgLugarAgregado),Toast.LENGTH_LONG).show()
44          } else {
45              Toast.makeText(requireContext(),getString(R.string.msgFaltanDatos),Toast.LENGTH_LONG).show()
46          }
47      }
48
49      private fun validos(nombre: String, correo: String, telefono: String, web: String): Boolean {
50          return !(nombre.isEmpty() || correo.isEmpty() || telefono.isEmpty() || web.isEmpty())
51      }
52
53  }
```

- Corremos el proyecto

fidélitas Universidad

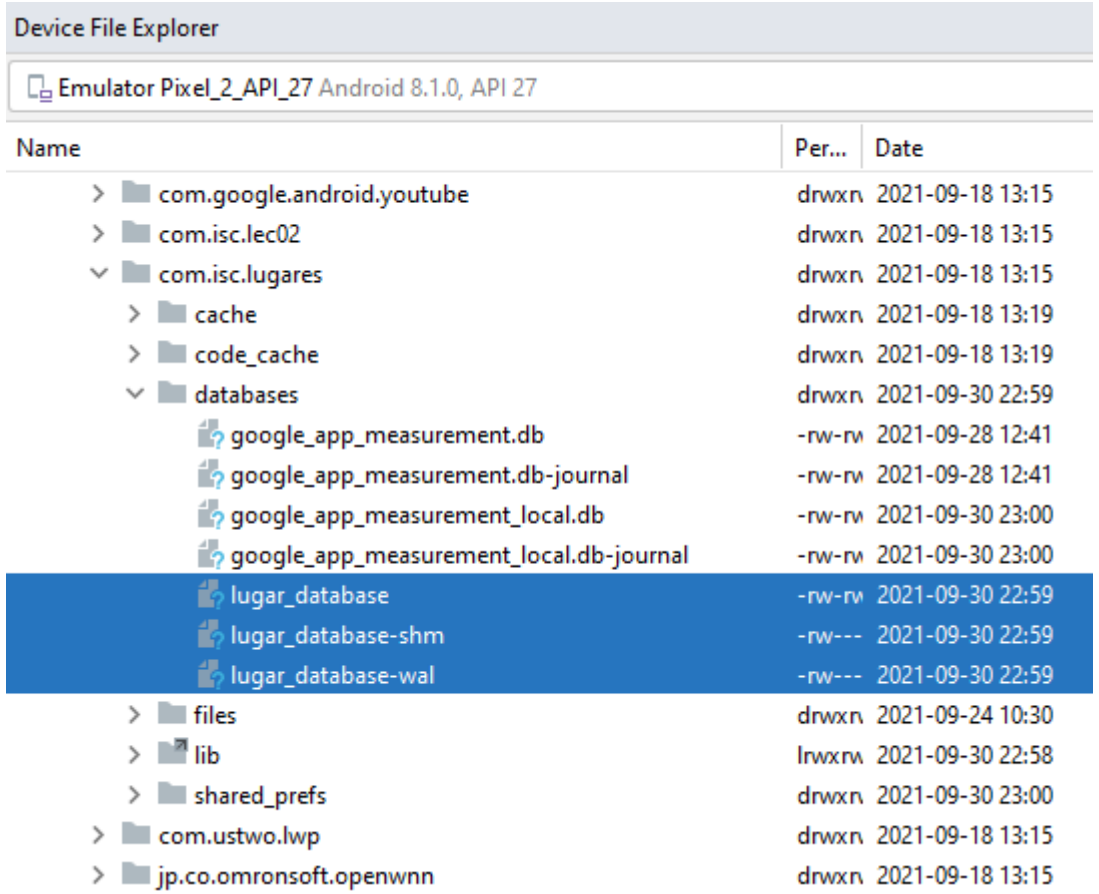


SE DESCARGA UN VISOR DE SQLITE

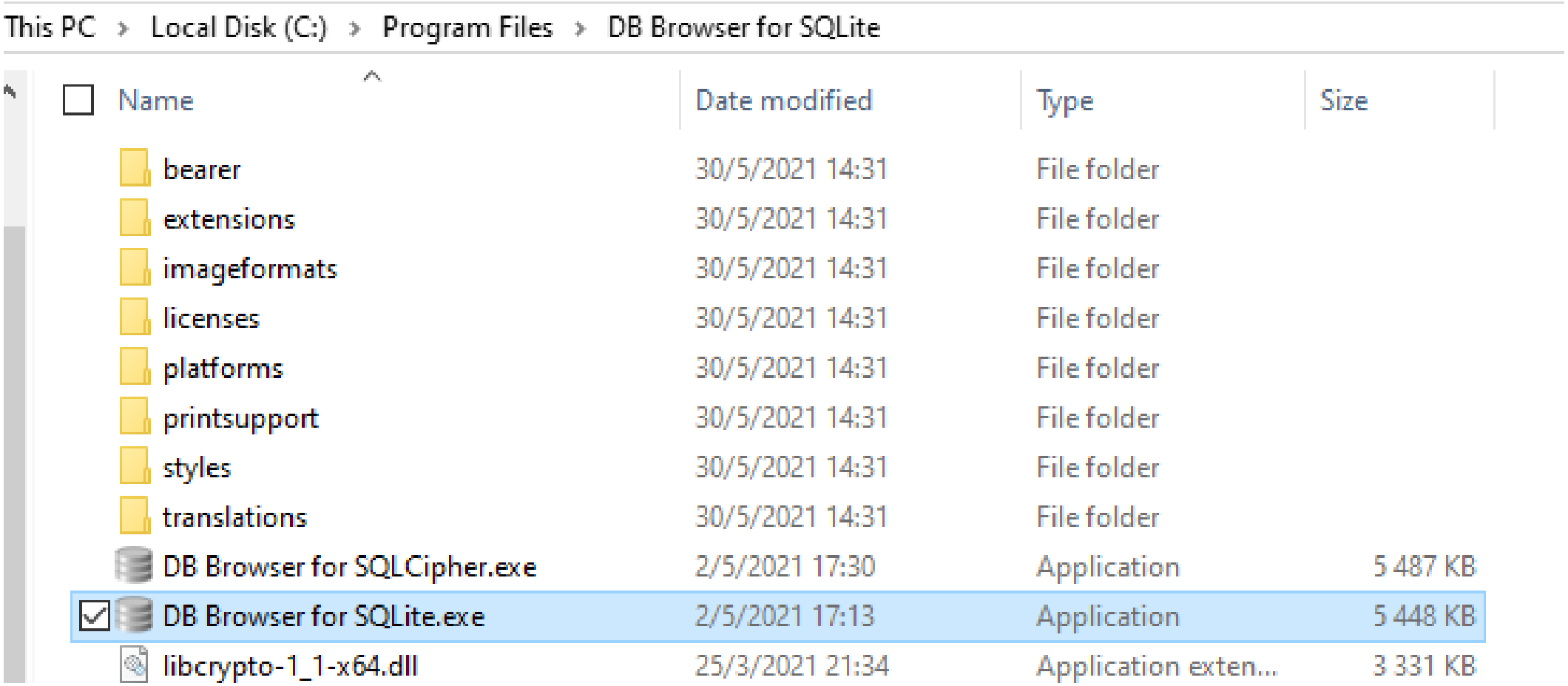
https://sqlitebrowser.org/dl/

SE COPIAN LOS TRES ARCHIVOS

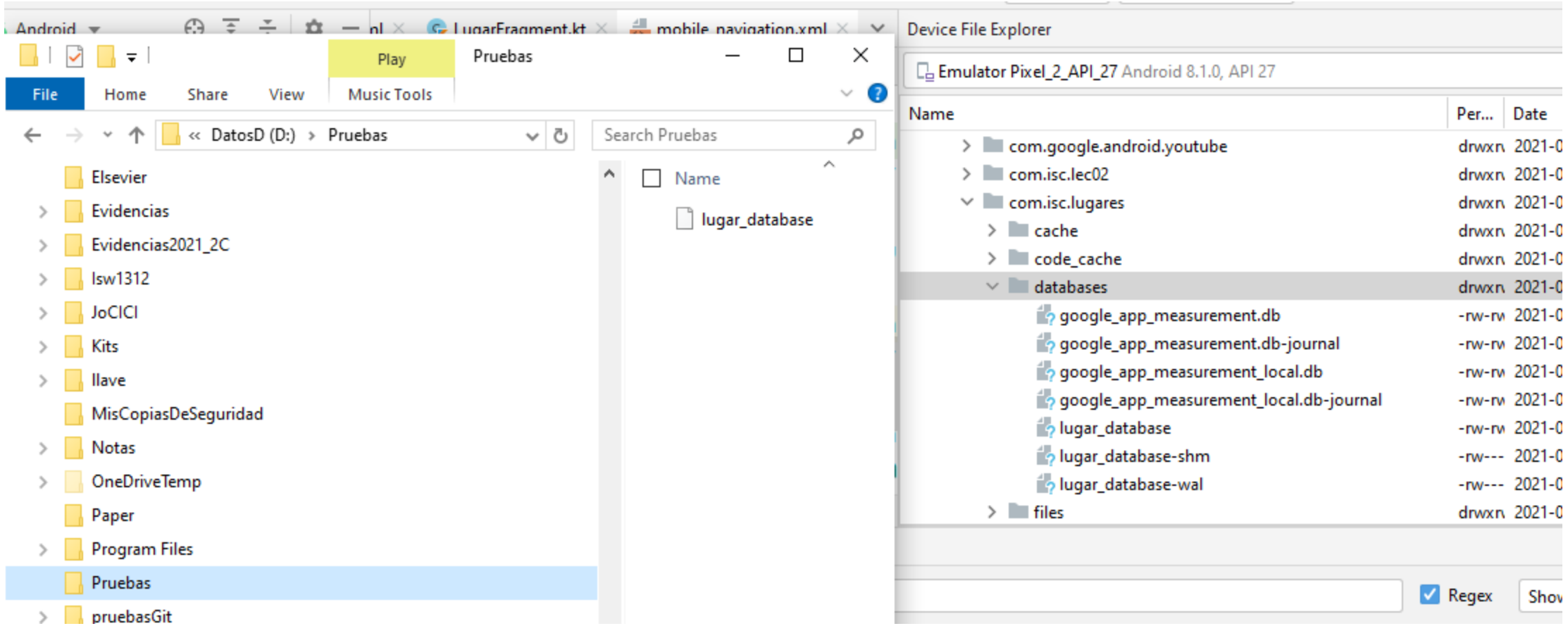
Se encuentran dentro de Device File Explorer, dentro de Data -> Data, se busca el paquete -> dentro del paquete está la carpeta databases



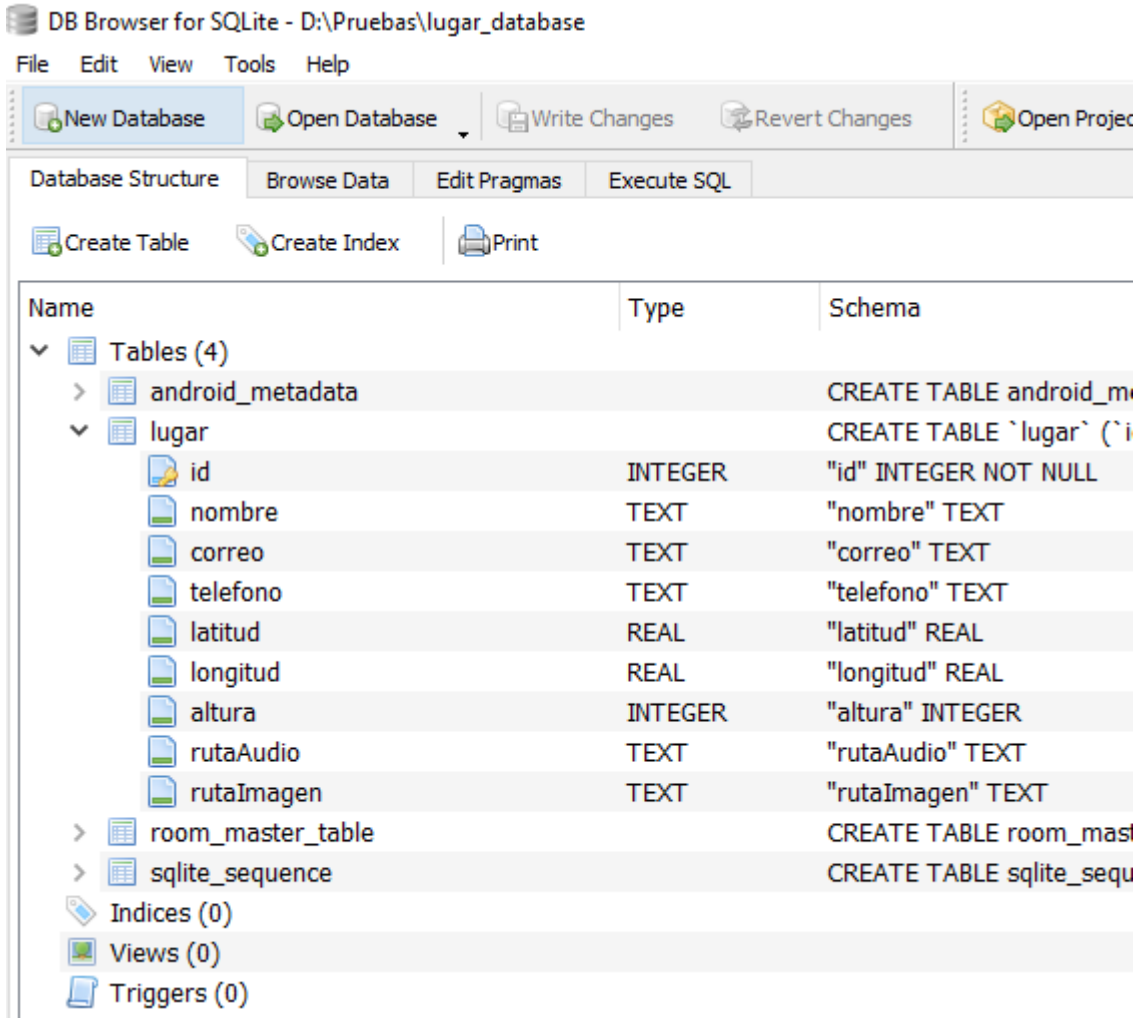
SE EJECUTA EL VISOR

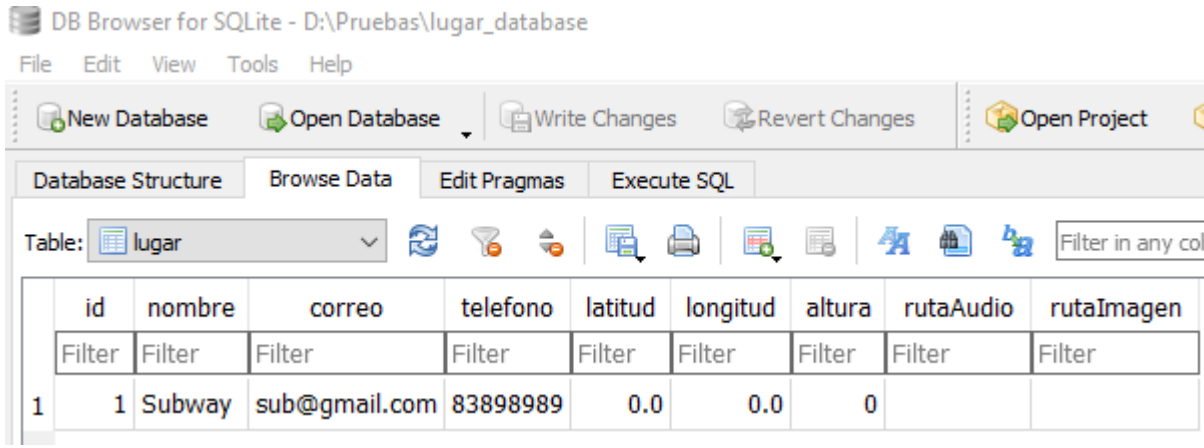


Se encuentran los archivos en el dispositivo y se copian en algún lugar



SE PUEDE APRECIAR LA INFORMACIÓN






CONTINUAMOS LA PRÓXIMA CLASE